Control and Real-time Scheduling Co-design : Application to Robust Robot Control

Daniel Simon, David Robert, Olivier Sename

Control and Real-time Scheduling Co-design Application to Robust Robot Control

Daniel Simon[1], David Robert[2], and Olivier Sename[2]

[1] INRIA Rhône-Alpes
655 avenue de l'Europe, Montbonnot
38334 Saint-Ismier Cedex, France
Daniel.Simon@inrialpes.fr
[2] Laboratoire d'Automatique de Grenoble
ENSIEG-BP 46
38402 Saint Martin d'Hères Cedex, France
{david.robert,olivier.sename}@inpg.fr

Abstract. Control systems running on a computer are subject to timing disturbances coming from implementation constraints. Fortunately closed-loop systems behave robustly w.r.t. modelling errors and disturbances, and the controller design can be performed to explicitly enhance robustness against specific uncertainties. On one hand robustness in process controllers can be used to comply with weakly modelled timing uncertainties. On the other hand the principle of robust closed-loop control can also be applied to the real-time scheduler to provide on-line adaption of some scheduling parameters, with the objective of controlling the computing resource allocation. The control performance specification may be set according to both control and implementation constraints. The approach is illustrated through several examples using simulation and an experimental feedback scheduler is briefly described.

1 Introduction

Digital control systems can be implemented as a set of tasks running on top of an off-the-shelf real-time operating system (RTOS) using fixed-priority and preemption. The performance of the control, e.g measured by the tracking error, and even more importantly its stability, strongly relies on the values of the sampling rates and sensor-to-actuator latencies (the latency we consider for control purpose is the delay between the instant when a measure q_n is taken on a sensor and the instant when the control signal $U(q_n)$ is received by the actuators [1]). Therefore it is essential that the implementation of the controller respect an adequate temporal behaviour to meet the expected performance. However implementation constraints such as multi-rate sampling, preemption, synchronisation and various sources of delays makes the run-time behaviour of the controller very difficult to accurately predict. However as we deal with closed-loop controllers we may take advantage of the *robustness* of such systems to design and implement flexible and adaptive real-time control architectures.

This paper deals with robust and adaptive solutions for real-time scheduling and control co-design. In the next section we review some properties of closed-loop controllers in contrast with real-time implementation constraints. Some recent results in control and scheduling co-design are recalled in section 3. Section 4 gives an overview of a new feedback scheduling strategy aimed to on-line adapt the tasks period according to the computing resource activity. This approach is then applied in the design of a robot controller in section 5, for which the importance of integrated control/scheduling co-design is emphasised. Finally an experimental feedback scheduler implementation is described in section 6 and future research directions conclude the paper.

2 Control and Implementation Constraints

Closed-loop digital control systems use a computer to periodically sample sensors, compute a control law and send control signals to the actuators of a continuous time physical process. The control algorithm can be either designed in continuous time and then discretized or directly synthesised in discrete time taking account of a model of the plant sampled by a zero-order holder. Control theory for linear systems sampled at fixed rates has been established a long time ago, e.g. [1].

Assigning an adequate value for the sampling rate is a decisive duty as this value has a direct impact on the control performance and stability. While an absolute lower limit for the sampling rate is given by Shannon's theorem, in practise rules of thumb are used to give a useful range of control frequencies according to the process dynamics and to the desired closed-loop bandwidth (see for example section 5.3). A general rule is that decreasing the control period and latencies allows for improved control performance, e.g. measured by the tracking error or disturbances rejection.

2.1 Digital Control of Continuous Systems

To implement a controller the basic idea consist in running the whole set of control equations in a unique periodic real-time task which clock gives the controller sampling rate. In fact all parts of the control algorithm do not have an equal weight and urgency w.r.t. the control performance. To minimise the latency a control law can be basically implemented as two real-time blocks, the urgent one sends the control signal directly computed from the sampled measures while updating the state estimation or parameters can be delayed or even more computed less frequently [1].

In fact, a complex system involves sub-systems with different dynamics which must be further coordinated [2]. Assigning different periods and priorities to different blocks according to their relative weight allows for a better control of critical latencies and for a more efficient use of the computing resource [3]. However in such cases finding adequate periods for each block is out of the scope

of current control theory and must be done through case studies, simulation and experiments.

For example [4] uses off-line iterative optimisation to compute an adequate setting of periods, latencies and gains according to the requested control performance and to the implementation constraints.

Latencies have several sources: the first one comes from the computation duration itself, and worst case execution times are difficult to get. In a multi-tasking systems they come from preemption due to concurrent tasks with higher priority, from precedence constraints and from synchronisation. Another source of delays is the communication medium and protocols when the control system is distributed on a network of connected devices. In particular it has been observed that in an synchronous multi-rate systems the value of sample-induced delays show complex patterns and can be surprisingly long [5].

2.2 Control and Timing Uncertainty

While timing uncertainties have an impact on the control performance they are difficult to be accurately modelled or constrained to lie inside precisely known bounds. Thus it is worth examining the sensitivity of control systems w.r.t. timing fluctuations.

Control systems are often cited as examples of "hard real-time systems" where jitter and deadline violations are strictly forbidden. In fact experiments show that this assumption may be false for closed-loop control. Any practical feedback system is designed to obtain some stability margin and robustness w.r.t. the plant parameters uncertainty. This also provides robustness w.r.t. timing uncertainties: closed-loop systems are able to tolerate some amount of sampling period and computing delays deviations, jitter and occasional data loss with no loss of stability or integrity, e.g. [6]: their behaviour can still be considered as correct as long as the sample-induced disturbances stay inside the performance specification bounds.

Therefore the hard real-time assumption can softened to better cope with the reality of closed-loop control. For example they can be changed for "weakly hard" constraints: absolute deadlines are replaced by statistical ones, e.g. the allowable output jitter compliant with the desired control performance or the number of allowed deadlines miss over a specified time window [7]. Note that to be fully exploited weakly hard constraints should be associated with a decisional process: tasks missing their deadline can be for example delayed, aborted or skipped according to their impact on the control law behaviour.

Finding the values of such weakly hard constraints for a given control law is currently out of the scope of current control theory in the general case. However the intrinsic robustness of closed-loop controllers allows for complying with softened timing constraints specification and flexible scheduling design.

2.3 Control and Scheduling

Usually, real-time systems are modelled by a set of recurrent tasks assigned to one or several processors and a worst case response times technique is used to analyse fixed-priority real-time systems. Well known scheduling policies, such as Rate Monotonic for fixed priorities and EDF for dynamic priorities, assign priorities according to timing parameters, respectively sampling periods and deadlines. They are said to be "optimal" as they maximise the number of tasks sets which can be scheduled with respect of deadlines, under some restrictive assumptions. Unfortunately they are not optimised for control purpose.

They hardly take into account precedence and synchronisation constraints which naturally appear in a control algorithm. The relative urgency or criticality of the control tasks can be unrelated with the timing parameters. Thus, the timing requirements of control systems w.r.t. the performance specification do not fit well with scheduling policies purely based on schedulability tests. It has been shown through experiments, e.g. [6], that a blind use of such traditional scheduling policy can lead to an inefficient controller implementation; on the other hand a scheduling policy based on application's requirements, associated with a right partition of the control algorithm into real-time modules may give better results.

Another example of unsuitability between computing and control requirements arises when using priority inheritance or priority ceiling protocols to bypass priority inversion due to mutual exclusion, e.g. to ensure the integrity of shared data. While they are designed to avoid dead-locks and minimise priority inversion lengths, such protocols jeopardise at run-time the initial schedule which was carefully designed to meet control requirements. As a consequence latencies along some control paths can be largely increased leading to a poor control performance or even instability.

Finally off-line schedulability analysis rely on a right estimation of the tasks worst case execution time. Even in embedded systems the processors use caches and pipelines to improve the average computing speed while decreasing the timing predictability. Another source of uncertainty may come from some pieces of the control algorithm. For example, the duration of a vision process highly depends on incoming data from a dynamic scene. Also some algorithms are iterative with a badly known convergence rate, so that the time before reaching a predefined threshold is unknown (and must be bounded by a timeout). In a dynamic environment some control activities can be suspended or resumed and control algorithms with different costs can be scheduled according to various control modes leading to large variations in the computing load.

Thus real-time control design based on worst case execution time, maximum expected delay and strict deadlines inevitably leads to a low average usage of the computing resource.

3 Related Work

Control/Scheduling Co-design This mainly concerns the integration of control performance knowledge in the scheduling parameters assignment. Indeed, once a control algorithm has been designed, a first job consists in assigning timing parameters, i.e. periods of tasks and deadlines, so that the controller's implementation satisfies the control objective. This may be done off-line or on-line.

In off-line control/scheduling co-design setting adequate values for the timing parameters rapidly falls into case studies based on simulation and experiments. For instance in [4] off-line iterative optimisation is used to compute an adequate setting of periods, latencies and gains resulting in a requested control performance according to the available computing resource and implementation constraints. Also in [8] the temporal requirements of the control system are described using complex temporal attributes (e.g. nominal period and allowed variations, precedence constraints...): this model is then used by an off-line iterative heuristic procedure to assign the scheduling parameters (e.g. priorities and offsets) to meet the constraints.

Concerning co-design for on-line implementation, recent results deal with varying sampling rates in control loops in the framework of linear systems: for example [9] show that, while switching between two stable controllers, too frequent control period switches may lead to unstability. Unfortunately most real-life systems are non-linear and the extrapolation of timing assignment through linearising often gives rough estimations of allowable periods and latencies or even can be meaningless. In fact, as it will be shown in the robot control application, the plant knowledge is necessary to get an efficient control/scheduling co-design.

Feedback Scheduling Besides traditional assignment of fixed scheduling parameters more flexible scheduling policies have been investigated. Let us cite e.g. [10] where the elasticity of the tasks' periods enables for controlling the quality of service of the system as a function of the current estimated load. While such an approach is still working in open loop w.r.t. a controlled plant, the on-line combination the control performance and implementation constraints lead to the feedback scheduling approach.

This new approach has been initiated both from the real-time computing side [11] and from the control side [12–14]. The idea consists in adding to the process controller an outer sampled feedback loop ("scheduling regulator") to control the scheduling parameters as a function of a QoC (Quality of Control) measure. It is expected that an on line adaption of the scheduling parameters of the controller may increase its overall efficiency w.r.t. timing uncertainties coming from the unknown controlled environment. Also we know from control theory that closing the loop may increase performance and robustness against disturbances when properly designed and tuned (otherwise it may lead to instability).

Figure 1 gives an general overview of a feed-back scheduler where an outer loop (the *scheduling controller*) adapts in real-time the scheduling parameters from measurements taken on the computer's activity, e.g. the computing load .

Besides this controller working periodically (at a rate larger than the sampling periods of the plant control tasks), the system's structure may evolve along a discrete time scale upon occurrence of events, e.g. for new tasks admission or exception handling. These decisional processes may be handled by another real-time task, the *scheduling manager*, which is not further detailed in this paper. Notice that such a manager may give a reference to the controller resource utilisation.

The design problem can thus be stated as control performance optimisation under constraint of available computing resources. Major studies result from [15, 6] where it is suggested that a simple solution to this optimal control problem (i.e. under resource constraint) is the calculation of the new task periods by the rescaling:

$$h_i^{new} = h_{i_{nominal}} \frac{U}{U_{sp}}$$

where U_{sp} is the utilisation set-point. The feedback scheduler then controls the processor utilisation by assigning task periods that optimise the overall control performance.

Preliminary works have been done by the authors. In [16] an LQG approach is used to design the feedback scheduling while in [17], an H_∞ control problem is solved for a two tasks systems (without differentiation between both tasks). In what follows the proposed methodology is described.

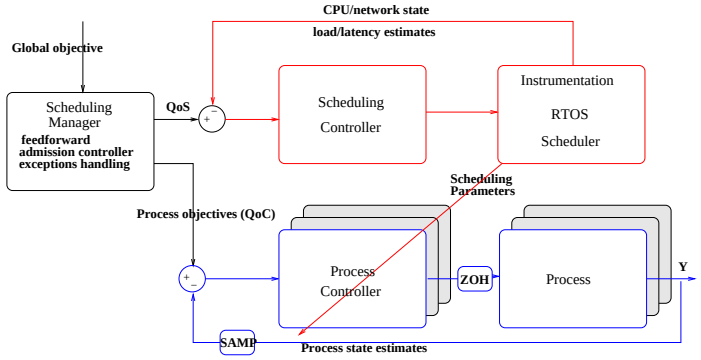

Fig. 1. Hierarchical control structure

4 A Robust Methodology for Feed-back Scheduling

Feedback scheduling is a dynamic approach allowing to better use the computing resources, in particular when the workload changes e.g. due to the activation of an admitted new task. Indeed, the CPU activity will be controlled according to the resource availability by adjusting scheduling parameters (i.e. period) of the plant control tasks.

In the approach here proposed, a way to take into account the resource sharing over a multitasks process is developed. In what follows, the control design issue is described including the control structure, the specification of control inputs and measured outputs, as well as the modelling step.

4.1 Control Structure

In Fig 2 scheduling is viewed as a dynamical system between control task frequencies and processor utilisation. As far as the adaptation of the control tasks is concerned, the load of the other tasks is seen as an output disturbance.

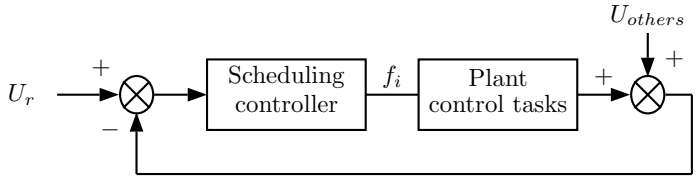

Fig. 2. Feedback scheduling bloc diagram

4.2 Sensors and Actuators

As stated in section 2.3, priorities must be assigned to control tasks according to their relative urgency ; this ordering remains the same in the case of a dynamic scheduler. Dynamic priorities, e.g. as used in EDF, only alter the interleaving of running tasks and will fail in adjusting the computing load w.r.t. the control requirements. In consequence we have elected the tasks periods to be the main actuators of the system running on top of a fixed priority scheduler[3].

As the aim is to adjust on-line the sampling periods of the controllers in order to meet the computing resource requirements, the control inputs are thus the periods of the control tasks.
The measured output is the CPU utilisation. Let us first recall that the scheduling is here limited to periodic tasks. In this case the processor load induced by a task is defined by $U = \frac{c}{h}$ where c and h are the execution time and period of the task. Hence processor load induced by a task is estimated, in a similar to way [14], for each period h_s of the scheduling controller, as :

$$\hat{U}_{kh_s} = \lambda\, \hat{U}_{(k-1)h_s} + (1-\lambda)\, \frac{\overline{c}_{kh_s}}{h_{(k-1)h_s}} \tag{1}$$

[3] Possible secondary actuators are variants of the control algorithms, with different QoS contributions to the whole system. Such variants should be handled by the scheduling manager working on a discrete events time scale

where h is the sampling frequency currently assigned to the plant control task (i.e. at each sampling instant kh_s) and $\overline{c}$ is the mean of its measured job execution-time. λ is a forgetting factor used to smooth the measure.

4.3 Control Design and Implementation

The proposed control design method for feedback scheduling is here developed. First one should note that, as shown in [17], if the execution times are constant, then the relation, $U = \sum_{i=1}^{n} C_i f_i$ (where $f_i = 1/h_i$ is the frequency of the task) is a linear function (while it would not be as a function of the task periods). Therefore, using (1), the estimated CPU load is given as:

$$\hat{U}(kh_S) = \frac{(1-\lambda)}{z-\lambda} \sum_{i=1}^{n} \overline{c}_i(kh_S) f_i(kh_S) \tag{2}$$

As $\overline{c}$ depends on the runtime environment (e.g. processor speed) a "normalised" linear model of the task i (i.e independent on the execution time) , G_i, is used for the scheduling controller synthesis where $\overline{c}$ is omitted and will be compensated by on-line gain-scheduling $(1/\overline{c})$ as shown below.

$$G_i(z) = \frac{\hat{U}(z)}{f_i(z)} = \frac{1-\lambda}{z-\lambda}, \quad i = 1, \ldots, n \tag{3}$$

As illustration, in a single control task system, the control scheme is therefore as in figure 3 where the estimated execution-times are used on-line to adapt the gain of the controller for the original CPU system (2) (this allows to compensate the variations of the job execution time).

According to this control scheme, the design of the controller K can be made using any advanced control methodology. For the considered application (see section 5), we have chosen the well known H_∞ control theory which can lead to a robust controller w.r.t modelling errors (see [18] for details on H_∞ control). Moreover it provides good properties in the presence of external disturbance, as it is emphasised in the illustrative examples.

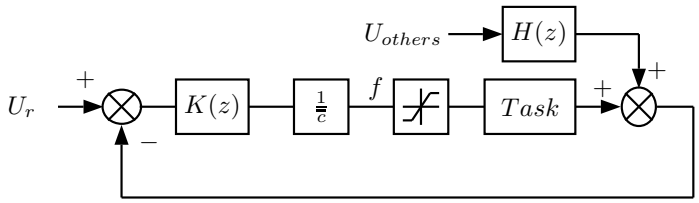

Fig. 3. Control scheme for CPU resources

5 Integrated Control/Scheduling Co-design in Robot Control

We consider here a seven degrees of freedom Mitsubishi PA10 robot arm that has been previously modelled and calibrated [19].

5.1 Plant Modelling and Control Structure

The problem under consideration is to track a desired trajectory for the position of the end-effector. Using the Lagrange formalism the following model can be obtained:

$$\Gamma = M(q)\ddot{q} + Gra(q) + C(q, \dot{q}) \tag{4}$$

where q stands for the positions of the joints, M is the inertia matrix, Gra is the gravity forces vector and C gathers Coriolis, centrifugal and friction forces.

The structure of the (ideal) linearising controller includes a compensation of the gravity, Coriolis/centrifugal effect and Inertia variations as well as a Proportional-Derivative (PD) controller for the tracking and stabilisation problem, of the form:

$$\Gamma = Gra(q) + C(q, \dot{q}) + K_p(q_d - q) + K_d(\dot{q_d} - \dot{q}), \tag{5}$$

leading to the linear closed-loop system $M(q)\ddot{q} = K_p(q_d - q) + K_d(\dot{q_d} - \dot{q})$.

This controller is divided in four tasks, i.e. a specific task is considered for the PD control, for the gravity, Inertia and Coriolis compensations, in order to use a multi-rate controller. In this first cautious feedback scheduling scheme, only the periods of the compensation tasks will be adapted, as they have a moderate impact on the closed-loop stability while they are more time consuming compared with the PD task.

5.2 Cost Functions

In the co-design, the aim is to give the precedence (i.e. more resources) to the tasks that are more important for the robot control. To evaluate this importance a cost function is defined as

$$J = \int_0^{t_{trajectory}} \sum_{i=1}^{i=7} (q_i - qd_i)^2 dt, \tag{6}$$

with q_i and qd_i the actual and desired positions of i^{th} joint. This cost function of figure 4 is here calculated off line first, for each subtask (among three) of the controller, according to the variation of the task periods (which may vary here from $0.5ms$ to $30ms$), the others remaining constant and fixed to $1ms$. Notice that we have chosen to let the PD control task at a constant period, due to its high influence in the robot control stabilisation strategy. As done in [16], a variable period could have been assigned to this task to emphasise its

importance in the closed-loop stability. However, in practical robot applications, it will remain at a constant rate.

Based on this cost functions it appears that gravity compensation is the more important task therefore we have to allocate it more resources. Costs of Coriolis and inertia compensation are quite similar thus gravity compensation resources allocation is chosen to be twice of Coriolis or inertia ones.

In this application, the period of the feedback scheduler has been fixed to $30ms$ to be larger than the robot control tasks (which limits have been set here from $0.5ms$ to $30ms$).

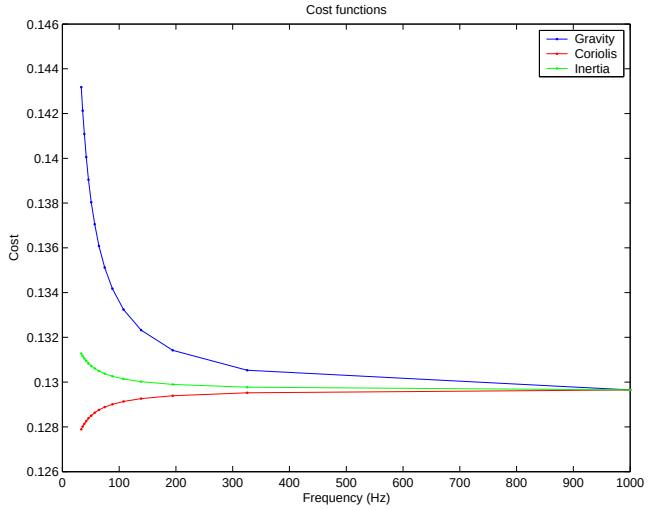

Fig. 4. Cost functions

5.3 Scheduling Controller Design

The bloc diagram of figure (5) is considered for the H_∞ design where $G'(z)$ is the model of the scheduler, the output of which is the vector of all task loads. To get the sum of all task loads, we have $C' = [1\ 1\ 1]$. $H(z)$ represents the sensor dynamical behaviour which measures the load of the other tasks. It may be a first order filter. The template W_e specifies the performances on the load tracking error as follows :

$$W_e(s) = \frac{s/M_s + \omega_b}{s + \omega_s \epsilon} \tag{7}$$

with $M_s = 2$, $\omega_s = 10\, rad/s$, $\epsilon = 0.01$ to obtain a closed-loop settling time of $300\, ms$, a static error less than $1\ \%$ and a good robustness margin. Matrix

$M = [1\ -1\ -1]$ and template W_x allow to specify the load allocation between the control tasks. With a large gain in W_x, it leads to:

$$U_{gravity} \approx U_{Coriolis} + U_{inertia},$$

i.e. we allocate more resources for the gravity compensation.

All templates are discretized with a sampling period of 30 ms. Finally discrete-time H_∞ synthesis algorithm produces a discrete-time scheduling controller of order 4.

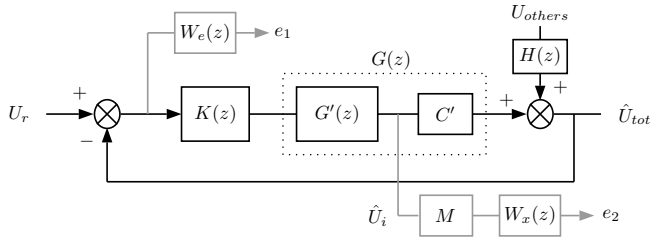

Fig. 5. H_∞ design bloc diagram

5.4 Simulations

Simulations are performed using the TrueTime toolbox presented in [20]

Benchmark: The trajectory to be tracked consists in a point to point motion, coming from the position $[-\pi/2, -\pi/2, -\pi/2, -\pi/2, -\pi/2, -\pi/2, -\pi/2]$ to $[\pi/2, 0, \pi/2, 0, \pi/2, 0, \pi/2]$ in the joint space at a constant velocity for each joint. The trajectory duration is set to 2 secs, thus it is slow enough to avoid reaching the actuators limits. The system is observed on a total duration of 2.5 seconds.

Concerning feedback scheduling, the initial set point of the resource utilisation is 60%. At $t = 1.5s$, to make room for an additionnal activity requesting admission, the resource utilisation set point is decreased to 30% of the full resource usage.

Result analysis : From Fig. 6 (periods) the admission9n of the disturbing task makes a transient increase of the processor load at $t = 1.5s$. To reject this disturbance, the scheduling controller increases the periods of the three compensators. On Fig. 6 (loads) it appear that the gravity compensation load is twice those of the inertia or Coriolis load as specified through the M and W_x templates.

On Fig. 6 (Angle) we may check that the control load variations have no noticeable effects on the trajectory (as expected due to the co-design and feedback scheduling strategies), whereas the commands appear to be a little more noisy on Fig. 6 (torques).

Remark 1 *As proposed in [16], the internal process controller can be also designed to take into account timing uncertainties, e.g. the control delays due to preemptions which are unavoidable in real-time control and difficult to accurately predict in a dynamic environment.*

6 Implementation of Feedback Schedulers

While TrueTime is very useful for fast prototyping and evaluation of scheduling controllers, it remains a simulation tool where the hardware and operating systems are abstracted. Execution times for control tasks are given as arbitrary values and execution times for the scheduler and system level housekeeping tasks are neglected. Therefore to assess the feasibility and practical interest of such adaptive closed-loop scheduling we have developed a feedback scheduler prototype on top of an existing robot controller. The current prototype runs under RTAI, the Realtime Linux Application Interface for Linux (`http://www.aero.polimi.it/~rtai/`). This RTOS provides an efficient real-time scheduler (with a recorded interrupt latency range of $1 - -7\mu sec$ with the Pentium II 450 MHz used for this setup) and allows time-stamping of events with the resolution of the built-in Time Stamp Counter (TSC), thus the execution time of each real-time tasks can be accurately measured and plotted.

The original controller uses the so-called *Computing Torque Controller* which is split into several computing modules to implement a multi-rate controller as in [3] (figure 7):

- CompTorque *is* the controller, updating the control torque vector U from the state (position and velocity) error vector $Q_d - Q$ using a PD algorithm which gains are tuned to provide a comfortable stability margin. The sampling rate and latency of this block have a strong impact on the system's performance and stability. The extra inputs coming from the robot model's computations are used to provide feed-forward; in the previous simulation framework, it is called the PD control task. As well, the period of this task is constant for stability requirements.
- Gravity, Coriolis and Inertia compute an explicit model of the robot arm's dynamics. They are costly to compute but it has been shown through simulations that they can be executed several times slower than the main controller with a moderate impact on the control performance and stability;
- GeneTraj provides the desired trajectory via-points at a fixed rate;
- The behaviour of the periodic control modules is supervised by a event driven reactive task in charge of setting up the system and exception handling, e.g. initialising the real-time tasks and cleanly stopping the system when it reaches the nominal termination state or when the control error exceeds a predefined threshold.

Control modules with different sampling rates communicate through asynchronous protected shared memories ; they are infinite loops triggered by clocks derived from the built-in timer by a middleware clock generator task (ClockGen).

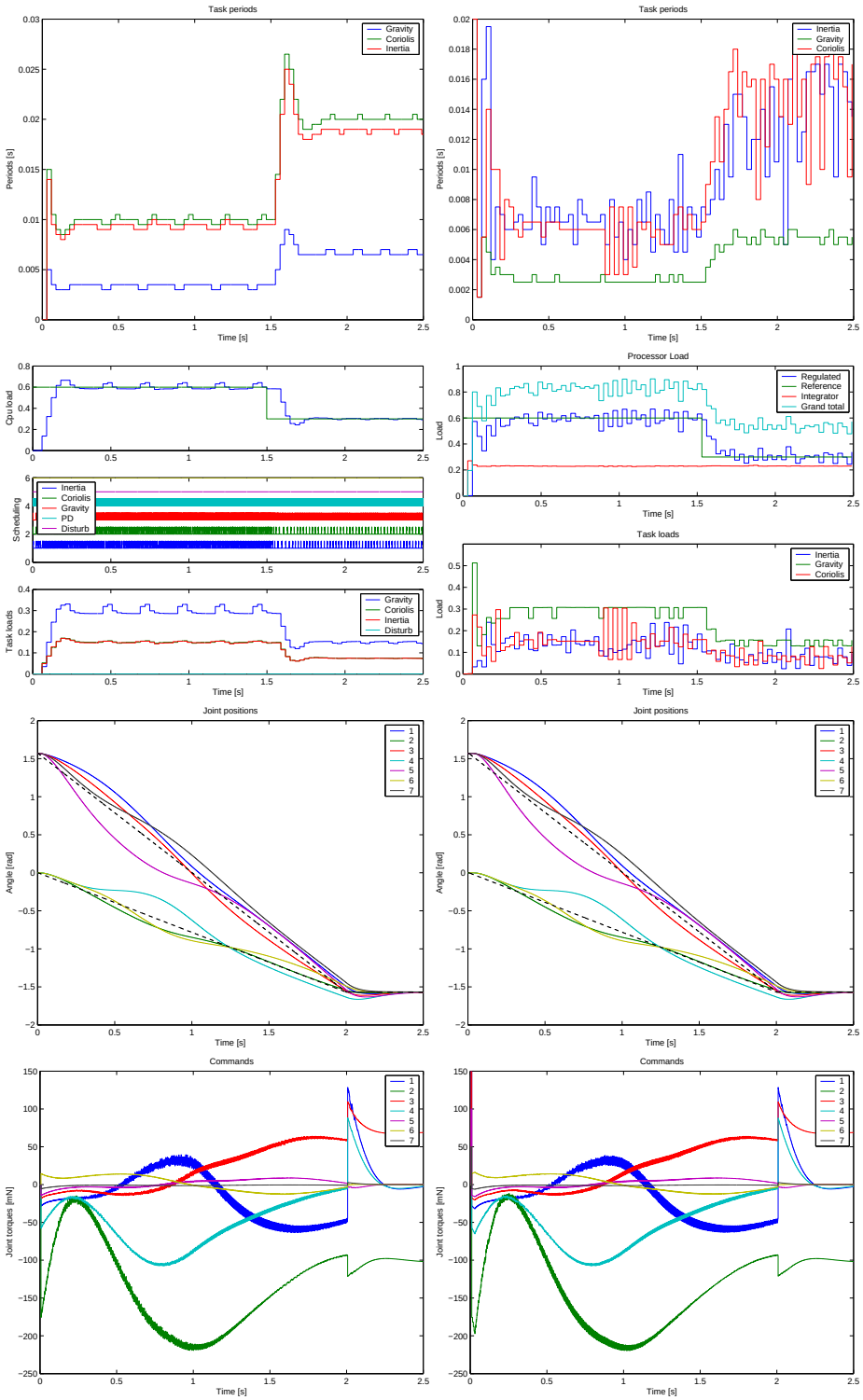

Fig. 6. Simulations with TrueTime (left) vs. execution under Linux/RTAI (right)

As our goal is testing the feasibility of feedback-scheduling using only the existing features of the operating system, the scheduling feedback loop is implemented as follows:

- The feedback scheduler is implemented as an additional real-time periodic task, i.e. a control module which function is specified and encoded by the control designer. The inputs are the measured execution times of the control tasks. The set point is a desired global computing load. Outputs are the sampling periods of the Gravity, Coriolis and Inertia control tasks;
- The period of CompTorque is fixed in this particular experiment. It runs at a fixed period of 1ms so that the stability of the system can be preserved. It implements the algorithm described in section 5.1 and reads the last available outputs of Gravity, Coriolis and Inertia via protected shared memories;
- The scheduling regulator adapts the periods of the Gravity, Coriolis and Inertia compensation modules and their computing load must be evaluated. The evaluation of the computing load is more or less easy and precise according the features of the operating system in use.
  On one hand, the direct measure of tasks execution times is, up to now, out of the scope of the API of most POSIX systems (such Linux and NPTL threads library). In that case, the computing load could be evaluated via the *time of response* of the tasks, but as the preemption phases cannot be discarded the global load is over-estimated. However POSIX-1003.1 defines a thread-specific CPU-time clock CLOCK_THREAD_CPUTIME_ID which can be used to record the threads activity (execution times or response time, according to the implementation of the library).
  On the other hand, the RTAI kernel keeps track of the *rt_tasks* execution cycles, thus allowing to recover a precise measure of the control tasks execution times from the scheduling regulator.
  Deadlines misses are checked and reported to the supervisor ; they are not taken into account in this preliminary setup but may be used in future improvements of a load estimator, for anti-windup and overload handling;
- In this experiment the robot is still (cautiously) simulated: the drivers call a numerical integrator running the robot's dynamics model every time a measure is taken or a new control vector is sent by the CompTorque task. Therefore the controller and simulated process time scales are coherent, but the simulation adds unpredicted latencies in the loop. The corresponding measured added load is about 25% of the Pentium CPU used for the setup which is fortunately strongly over-sized for this robot control algorithm;
- Priorities are ordered according to the relative urgency and weight of these function blocks on the system's behaviour: $ClockGen \succ Supervisor \succ FedSched \succ CompTorque \succ GeneTraj \succ Gravity \succ Coriolis \succ Inertia$
- The initial periods has been set to 20 msec for the three tasks with a variable and controlled period, i.e. Gravity, Coriolis and Inertia. The PD control task is run at a fixed rate of 1ms and the trajectory generator provides new set points every 5ms. The feedback scheduler is run every 30 msec.

– The sampling frequency of the clock generator is set to 2 KHz thus allowing to increment or decrement the control tasks clocks by $500\mu sec$ steps. Observing and managing the system at a faster rate would induce a very high system's load due to too many context switches.
– The desired trajectory is the same as in 5. Preliminary experiments shown that the robot's numerical integration spends about 25% of the available CPU power. As we need some load margin to avoid transient overruns the desired load is initially set to 0.6. At time 1.5 sec it is decreased to 0.3 to make room for a disturbing incoming task proposed for admission.

As plot in figure 6 the first experimental results are encouraging: they show that such a feedback scheduling architecture can be quite easily designed and implemented on top of an off-the-shelf real-time operating system with fixed priority and preemption. Examining the plots and the kernel reporting file calls for the following comments:

– As the feedback scheduler is a simple feedback algorithm running at a slow rate its computing cost is quite low (about $75\mu sec$ every $30ms$). On the other hand observing and managing the system at a high rate can be very costly due to numerous context switches. A more efficient implementation is currently studied to decrease the system's cost.
– The tasks loads can be directly measured using a specific, non portable feature of the RTAI kernel. As response times are easier to measure (e.g.from a POSIX API), using an execution time estimator would increase the portability of the system. Designing such a low-cost and reliable estimator remains to be done.
– Overshoots in the control periods and load lead to transient overload and deadline misses. These events are not currently processed leading to an avalanche of timing faults and to a system unstability and failure. Overruns and other timing errors must be adequately processed at the supervision level to more safely manage the system and make use of the full range of available processing power.

7 Summary and Further Research

In this paper, a methodology for robust control and scheduling co-design is proposed. While robust control usually deals with modelling errors of the controlled plant, a digital closed-loop controller is also submitted to timing disturbances coming from its implementation using a real-time operating system. These disturbances are difficult to be accurately predicted, and studying the impact of timing deviations in feedback loops is still a largely unexplored domain. Hence a natural idea consists in using robust control theory to design controllers to be weakly timing sensitive. Besides process control this idea can be used also to design a feedback scheduling loop to implement robust on-line adaption of the scheduling parameters according to estimates of the computing activity. Thus

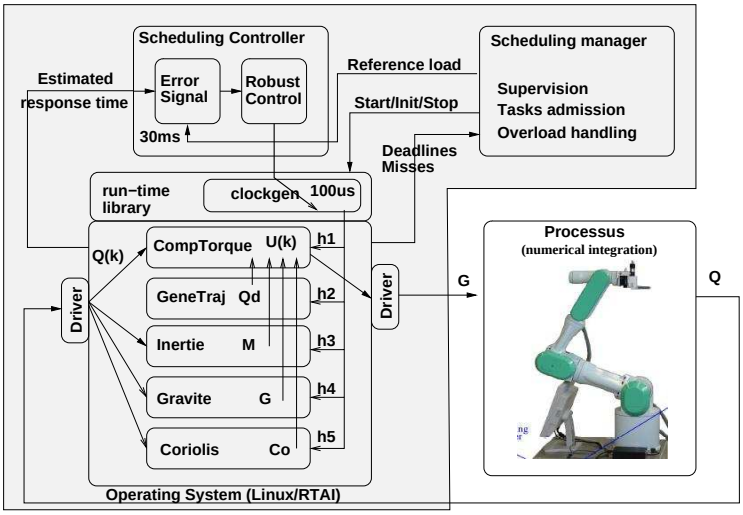

Fig. 7. Feed-back scheduling experiment

this resource allocation control loop is used to fulfil the plant control objective under constraint of limited computing resource.

An integrated control-scheduling framework is proposed. The control periods are weighted according to their impact on the control performance. An outer scheduling controller then regulates in real-time the CPU load according to the allocated computing power. Indeed the control synthesis of the feedback scheduler has been provided using the H_∞ control theory, and the gain of the controller is adapted on-line using the estimated execution times of the control tasks. In all cases the effectiveness of the controllers lies in a right choice of the design weighting functions used to specify the trade-off between concurrent constraints. Therefore the designer(s) must have knowledge on both the process and computing platform capabilities to be able to best fit the end-user's requirements.

Some simulation and experiment results have been given, which emphasises the interest of this approach and a software prototype has been designed to assess the feasibility of the approach using an off-the-shelf real-time operating system. The experiments show that the approach can be efficiently implemented with a moderate programming effort and a low or moderate real-time computing cost.

Further works firstly concern the improvement of robust control schemes for both the process and scheduler controllers. A better insight in control performance w.r.t. timing parameters is necessary to efficiently shape the weighting between control and computing constraints. Robustness of closed-loop control w.r.t. timing uncertainties must be investigated : in particular understanding the behaviour of controllers submit to deadlines miss or data loss deserves to be done, together with the examination of the links between a feedback sched-

uler and the underlying scheduling policy. Thus exception handling strategies for timing disturbances beyond the capabilities of robust feedback control could be implemented in a supervision layer which has been only sketched in the current experiments.

The feasibility of the implementation must also be taken into account. Using adaptive scheduling needs to carry out measures of the computing platform activity to be used at the control level; synthetic measurements must be made available in user space via a dedicated middleware layer or an extended API (e.g. in real-time POSIX). In case of a distributed control system the communication layer must also be instrumented according to the flexible scheduling and control needs.

Finally choosing strategies and tuning parameters leading to an effective trade-off which fit with the end-user's requirements needs a common understanding and cooperation between control and computer scientists and engineers.

References

1. Åström, K., Wittenmark, B.: Computer-Controlled Systems. 3rd edn. Information and systems sciences series. Prentice Hall, New Jersey (1997)
2. Törngren, M.: Fundamentals of implementing real-time control applications in distributed computer systems. Real Time Systems **14**(3) (1998) 219–250
3. Simon, D., Castillo, E., Freedman, P.: Design and analysis of synchronization for real-time closed-loop control in robotics. IEEE Trans. on Control Systems Technology **6**(4) (1998) 445–461
4. Ryu, M., Hong, S., Saksena, M.: Streamlining real-time controller design: from performance specifications to end-to-end timing constraints. In: IEEE Real Time Systems Symposium. (1997)
5. Wittenmark, B.: A sample-induced delays in synchronous multirate systems. In: European Control Conference, Porto, Portugal (2001) 3276–3281
6. Cervin, A.: Integrated Control and Real-Time Scheduling. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden (2003)
7. Bernat, G., Burns, A., Llamosí, A.: Weakly hard real-time systems. IEEE Transactions on Computers **50**(4) (2001) 308–321
8. Sandström, K., Norström, C.: Managing complex temporal requirements in real-time control systems. In: 9th IEEE Int. Conf. and Workshop on the Engineering of Computer-Based Systems (ECBS'02), Lund, Sweden (2002)
9. Schinkel, M., Chen, W.H., Rantzer, A.: Optimal control for systems with varying sampling rate. In: Proceedings of American Control Conference, Anchorage (2002)
10. Buttazzo, G., Abeni, L.: Adaptive rate control through elastic scheduling. In: 39th Conference on Decision and Control, Sydney, Australia (2000)
11. Lu, C., Stankovic, J.A., Tao, G., Son, S.H.: Feedback control real-time scheduling: Framework, modeling, and algorithms. Real Time Systems **23**(1) (2002) 85–126
12. Cervin, A., Eker, J.: Feedback scheduling of contol tasks. In: Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia (2000)
13. Eker, J., Hagander, P., Årzén, K.E.: A feedback scheduler for real-time control tasks. Control Engineering Practice **8**(12) (2000) 1369–1378
14. Cervin, A., Eker, J., Bernhardsson, B., Årzén, K.E.: Feedback-feedforward scheduling of control tasks. Real-Time Systems **23**(1) (2002)

15. Eker, J., Hagander, P., Årzén, K.E.: A feedback scheduler for real-time controller tasks. Control Engineering Practice **8**(12) (2000) pp 1369–1378
16. Sename, O., Simon, D., Robert, D.: Feedback scheduling for real-time control of systems with communication delays. In: ETFA'03 9th IEEE International Conference on Emerging Technologies and Factory Automation, Lisbonne (2003)
17. Simon, D., Sename, O., Robert, D., Testa, O.: Real-time and delay-dependent control co-design through feedback scheduling. In: CERTS'03 Workshop on Co-design in Embedded Real-time Systems, Porto, ECRTS (2003)
18. Zhou, K., Doyle, J.C., Glover, K.: Robust and optimal control. Prentice-Hall Inc. (1996)
19. Simon, D., Kapellos, K., Espiau, B.: Control laws, tasks and procedures with orccad: Application to the control of an underwater arm. Int. Journal of Systems Science **29**(10) (1998) 1081–1098
20. Henriksson, D., Cervin, A., Årzén, K.E.: Truetime: Simulation of control loops under shared computer resources. In: 15th IFAC World Congress on Automatic Control, Barcelona, Spain (2002)